Санкт-Петербургский государственный университет

Математическое обеспечение и администрирование информационных систем
Технология программирования

Ананьев Денис Владиславович

Разработка алгоритма определения языка для коротких текстов

Бакалаврская работа

Научный руководитель:
к. ф.-м. н., доцент Бугайченко Д. Ю.

Рецензент:
к. ф.-м. н., доцент Николенко С. И.

Санкт-Петербург
2016

SAINT-PETERSBURG STATE UNIVERSITY

Mathematical Software and Information Systems Administration
Technology of Programming

Denis Ananev

Implementation of language detection algorithm for short texts

Bachelor's Thesis

Scientific supervisor:
PhD Dmitry Bugaychenko

Reviewer:
PhD Sergey Nikolenko

Saint-Petersburg
2016

Оглавление

Введение

Возможность публикации собственного контента пользователями повлекла колоссальный рост данных с постоянно увеличивающейся скоростью. Зачастую такие данные являются неструктурированными, а их обработка должна происходить оперативно. Стало необходимым изобретение новых подходов, методов и инструментов для осуществления более эффективной обработки информации.

С другой стороны исследователей давно волновали вопросы, связанные с обработкой естественного языка, для организации более удобного взаимодействия между человеком и компьютером. Однако решение большинства задач данной области до определенного момента являлось затруднительным из-за недостаточного количества исходных данных.

Таким образом с прошествием времени, ввиду накопления необходимого количества информации и наличия исследовательского интереса, возник запрос на анализ имеющихся данных с целью выделения новых признаков для дальнейших разработок. Это позволило компаниям усовершенствовать свои сервисы для улучшения качества предоставляемых пользователям услуг. Стало возможным на основе предыдущего пользовательского опыта осуществлять рекомендации потенциально интересного контента. Качество же подобных рекомендаций все больше возрастает по мере развития методов машинного обучения.

В данной работе рассмотрена задача определения языка в коротких текстах. Подобная задача является одним из необходимых первых шагов для построения более сложных систем. Если имеется сервис, в котором пользователи могут публиковать свои сообщения на многих языках, то для рекомендации нового контента или формирования пользовательской новостной ленты в первую очередь следует определить языки сообщений, которые могут быть интересны пользователю. Ввиду отсутствия метаданных о тексте, касающихся содержащегося языка, эта задача представляет интерес.

Тривиальный подход подразумевает проверку входящих в сообщения слов на наличие в словарях конкретного языка. Однако такой под-

ход нецелесообразен с практической точки зрения. Для работы подобного метода требуется большое количество вычислительных мощностей и времени, но в реальных задачах необходимо осуществлять моментальную проверку языка и тратить на это как можно меньше ресурсов. Поэтому наилучшим образом себя зарекомендовал подход, основанный на построении вероятностной модели с помощью N-грамм символов [1]. Такой способ позволяет определять язык с весьма высокой точностью и достигнуть оптимальных показателей из соотношения затрат ресурсов и полученного качества, однако имеет некоторые недостатки.

Построение вероятностной модели на основе N-грамм не позволяет выделить достаточного количества признаков из коротких текстов, имеющих длину в несколько слов. Сообщения подобной длины зачастую встречаются в социальных сетях и сервисах микроблогов. Поэтому распознавание языка в текстах подобной длины представляет некоторые затруднения. Однако был предложен новый подход на основе использования всех содержащихся подстрок [5]. Такой способ был реализован в библиотеке ldig (Language Detection with Infinity Gram) [8]. В ней осуществляется поддержка 19 языков, основанных на латинском алфавите.

Таким образом представляет интерес разработка инструмента, поддерживающего расширенный набор языков. При этом необходимо учитывать специфику конкретных языков и их машинного представления, а также проводить предварительную нормализацию, для устранения зашумлённости и элементов, имеющих малый содержательный потенциал в отношении данной задачи.

1. Цель работы

Целью данной работы является построение алгоритма для определения языка в коротких текстах на основных государственных языках стран постсоветского пространства. Для этого необходимо решить следующие задачи:

- Исследовать существующие методы решения
- Подготовить необходимые корпусы текстов
- Провести нормализацию документов
- Построить соответствующие языковые модели
- Осуществить оценку качества полученного решения

2. Обзор существующих решений

В данном разделе будут рассмотрены существующие подходы к решению задачи определения языка.

2.1. Использование наиболее употребляемых слов

Для начала рассмотрим метод, основывающийся на вхождении определенных коротких слов, принадлежащих языку, предложенный в статье [2]. Конкретнее для каждого языка были использованы тексты длиной в миллион символов, из которых извлекались слова до пяти символов, появляющиеся в тексте не менее трех раз. Таким образом были получено большинство употребляемых коротких слов. В статье [6] показано, что подобный подход аналогичен использованию служебных слов языка.

В следующем методе [3] для построения языковой модели используется определенное количество наиболее часто встречающихся слов произвольной длины. Таким образом для каждого языка строится модель на основе слов, имеющих максимальную частоту вхождения в соответствующие ему документы. Количество таких слов различается в разных работах.

2.2. Language Detection Library

Одним из наиболее успешных решений задачи определения языка для текстов произвольной длины является библиотека lang-detect [7]. Поддержка большего количества языков представляет большее количество трудностей. Особенно затруднительным является определение похожих языков из одной языковой группы, использующих множество общих слов.

Основным методом данной библиотеки является формирование языковых профилей с помощью подсчета вероятностей признаков для документов данного языка. Для этого используется наивный байесовский классификатор с N-граммами символов.

Автор замечает, что само по себе построение необходимых языковых корпусов является затруднительной задачей, так как для многих редких языков не сущестует подходящих датасетов. Кроме того в существующих решениях поддерживается малое количество языков (в основном европейских), а также имеется низкая точность, затрудняющая использование в практических целях.

Таким образом в данной библиотеке поставлена задача достижения точности свыше 99%, поддержка 50 различных языков, представляющих различные языковые семьи, а также быстрый подсчет предсказания для конкретного документа. Профиль языка генерируется из размеченного тренировочного корпуса на основе Википедии. Работа библиотеки протестирована на 200 новостных статьях, показано достижение необходимой точности.

Процесс распознавания завершается при достижении точности свыше 0.99999. Производится подсчет количества N-грам с учетом разделителей, обозначающих начало и конец слова. Каждый язык имеет свойственные правила написания, а также наиболее часто встречающиеся последовательности символов. Аккумулируя вероятности соответствующих признаков, предполагаемый язык получается как результат, имеющий наибольшую вероятность.

Подобный вариант классификации имеет точность около 90%, а среди некоторых языков и куда более низкий результат. Причиной тому является большое количество шума как в тренировочных, так и в тестовых документах. Поэтому необходимо проводить фильтрацию шума. Кроме того для языков, имеющих алфавиты, значительно превосходящие размер латинского (например, китайский), свойственна проблема нулевой частоты. Тренировочные корпуса таких языков зачастую состоят из наиболее употребительных символов, состовляющих лишь малую часть.

Поэтому устранение шума в корпусах и документах является важным шагом. На этом этапе удаляются нерепрезентативные для языка символы (числа, ссылки и тп). Устраняется текст на латинице, часто использующийся в других языках (например, аббревиатуры). Также

нерепрезентативными являются слова, полностью состоящие из заглавных букв.

Таким образом библиотека достигает точности свыше 99.8% для большинства языков. Несмотря на высокую точность, имеются некоторые проблемы. Подобная точность достигается на текстах довольно большой длины. Однако в сети часто встречаются короткие пользовательские сообщения. Хорошим примером подобного сервиса является платформа микроблогов Twitter. Также открытыми проблемами являются распознавание сообщений на нескольких языках и нахождение в тексте программного кода. Проблема коротких сообщений была частично решена в дальнейшей разработке автора [8], описанной в следующей главе.

2.3. Language Detection with Infinity Gram

В данной работе автором был реализован новый подход с целью достижения точности свыше 99% для коротких текстов минимальной длиной более трех слов. Так как подобная длина является недостаточной для извлечения 3-грамм, был предложен алгоритм, основывающийся на извлечении признаков в виде всех подстрок произвольной длины [5]. Признаки хранятся в виде префиксного дерева. Вводится отношение включения, по которому подстроки разбиваются на классы эквивалентности. Каждый класс содержит уникальную максимальную подстроку. Все вхождения остальных элементов соответствующего класса происходят только в максимальной подстроке, то есть частоты подстрок, состоящих в отношении включения эквиваленты. Для классификации используется логистическая регрессия.

В данной работе также большое значение уделено подготовке и обработке необходимых текстовых корпусов. Список был ограничен 19 языками, представляющим наибольшую сложность по мнению автора. Было произведено множество нормализаций для конкретных языков. Таким образом была достигнута средняя точность и полнота предсказывания свыше 99%.

К недостаткам данной работы можно отнести весьма ограниченный список поддерживаемых языков.

3. Подготовка данных

Для построения языковых моделей необходимо подготовить данные в соответствующем формате. Для тренировки ldig текстовые корпусы необходимо собрать в единый файл, строки которого имеют вид **[label]\t[content]**. В качестве **content** используются текстовые сообщения на определенном языке. Длину таких сообщений будем ограничивать в 140 символов. В качестве **label** служит метка о языке сообщения. В частности нас интересуют основные государственные языки постсоветского пространства, весьма слабо отраженные в публикациях по соответствующей тематике. Метки данных языков в соответствии с ISO 639-1 имеют следующий вид:

- az – азербайджанский
- be – белорусский
- hy – армянский
- ka – грузинский
- kk – казахский
- ky – киргизский
- ro – румынский (официальный язык Республики Молдова)
- ru – русский
- tg – таджикский
- tk – туркменский
- uk – украинский
- uz – узбекский

Для большинства вышеперечисленных языков не существует подходящих готовых текстовых корпусов. Поэтому были использованы открытые данные Википедии. Для каждого из языков предоставлены соответствующие XML-файлы, содержащие контент и дополнительную метаинформацию. Поскольку для дальнейшего построения языковых профилей необходимыми являются только текстовые документы и метка языка, мы извлекаем основной контент XML-файлов. После чего призводится фильтрация полученных данных для устранения элементов, не релевантных для данной задачи. Устраняются пунктуационные знаки, числа, а также символы латинского алфавита для языков, использующих иную нотацию.

Так как библиотека ldig рассчитана на обработку сообщений короткой длины, происходит разбивка длинных сообщений на строки с ограничением в 140 символов, для каждой из которых указывается соответствующая языковая метка. Таким образом для каждого языка формируются текстовые файлы по 75000 сообщений, за исключением таджикского и туркменского, для которых представлено 58373 и 32855 сообщений соответственно.

4. Оценка работы моделей

В данном разделе приводятся оценки работы моделей с использованием N-грамм и ∞-грамм для стандартных языковых профилей, а также натренированных с помощью подготовленного корпуса. В таблице 1 отражены показатели для ldig и lang-detect, полученные с помощью 5-разовой кросс-валидации.

Для осуществления кросс-валидации имеющийся корпус разбивается на пять частей. После этого на четырёх частях проводится обучение, а на пятой — тестирование. Данная процедура проводится пять раз, в каждый из которых смещается тестирующая выборка. Таким образом можно получить различные показатели качества, а также замерить их доверительные интервалы для каждого класса.

Метрики измерения качества вычисляются следующим образом. После работы классификатора каждый из экземпляров можно отнести к одному из четырех классов:

- True Positive (TP) — экземпляр, правильно отнесенный к положительному классу
- False Positive (FP) — экземпляр, неправильно отнесенный к положительному классу
- True Negative (TN) — экземпляр, правильно отнесенный к одному из отрицательных классов
- False Negative (FN) — экземпляр, неправильно отнесенный к одному из отрицательных классов

В результате подобного разбиения можно измерить различные характеристики качества на полученной выборке. В данном случае нас будут интересовать точность (precision), полнота (recall) и f1-score. Они вычисляются следующим образом:

$$precision = \frac{TP}{TP + FP}$$

$$recall = \frac{TP}{TP + FN}$$

$$f1 - score = \frac{2TP}{2TP + FP + FN}$$

Доверительные интервалы считаются с учетом предположения о нормальном распределении с вероятностью 0.95.

Ввиду того, что стандартные профили библиотеки ldig содержат только часть языков латинского алфавита, из интересующих нас языков удалось измерить только румынский. Для библиотеки lang-detect получены результаты для белорусского, румынского, русского и украинского. Достигаются следующие показатели на подготовленном корпусе.

Из таблицы 1 можно заметить невысокие показатели точности lang-detect, достигнутые на тренировочном корпусе с использованием стандартных профилей. Это объясняется наличием в корпусе схожих кириллических языков, для которых отсутствуют необходимые профили. Поэтому многие сообщения были ошибочно отнесены в классы русского или белорусского языков. Более подробно матрица ошибок отражена в таблице 2.

Таким образом для повышения точности определения интересующих нас языков необходимо добавить недостающие языковые профили. В таблице 3 показано сравнение количества верно определенных представителей для lang-detect при использовании стандартных профилей с добавлением натренированных и при использовании только натренированных профилей.

язык	метрика	стандартные профили		натренированные профили	
		ldig	lang-detect	ldig	lang-detect
az	precision	–	–	0.9996±2.2e-4	0.9996±3.8e-4
	recall	–	–	0.9980±1.5e-3	0.9975±1.7e-3
	f1-score	–	–	0.9988±6.6e-4	0.9985±8.0e-4
be	precision	–	0.5230±4.5e-3	0.9948±5.7e-3	0.9998±1.1e-4
	recall	–	0.9942±9.7e-4	0.9975±8.7e-4	0.9953±3.9e-3
	f1-score	–	0.6854±4.0e-3	0.9961±2.7e-3	0.9976±1.9e-3
hy	precision	–	–	0.9987±2.0e-3	1.0000±5.6e-5
	recall	–	–	0.9986±4.9e-4	0.9976±2.0e-3
	f1-score	–	–	0.9987±8.2e-4	0.9988±1.0e-3
ka	precision	–	–	1.0000±0.0e+0	0.9999±5.6e-5
	recall	–	–	0.9995±2.1e-4	0.9992±4.4e-4
	f1-score	–	–	0.9998±1.4e-4	0.9996±2.4e-4
kk	precision	–	–	0.9929±9.6e-3	0.9993±7.4e-4
	recall	–	–	0.9954±1.0e-3	0.9945±1.7e-3
	f1-score	–	–	0.9941±5.0e-3	0.9969±8.5e-4
ky	precision	–	–	0.9957±4.8e-3	0.9994±4.1e-4
	recall	–	–	0.9921±2.3e-3	0.9879±5.7e-3
	f1-score	–	–	0.9939±2.0e-3	0.9936±3.0e-3
ro	precision	0.9874±5.2e-4	0.9994±1.0e-4	0.9981±8.7e-4	0.9646±3.1e-2
	recall	0.9951±6.7e-4	0.9965±1.9e-4	0.9991±3.0e-4	0.9996±2.1e-4
	f1-score	0.9912±3.1e-4	0.9980±6.8e-5	0.9986±3.2e-4	0.9817±1.6e-2
ru	precision	–	0.4591±3.8e-3	0.9730±6.0e-3	0.9660±1.7e-2
	recall	–	0.9962±4.6e-4	0.9775±2.8e-2	0.9968±8.8e-4
	f1-score	–	0.6285±3.6e-3	0.9751±1.2e-2	0.9811±9.2e-3
tg	precision	–	–	0.9722±3.0e-2	0.9982±1.3e-3
	recall	–	–	0.9690±6.2e-3	0.9450±5.0e-2
	f1-score	–	–	0.9705±1.3e-2	0.9705±2.6e-2
tk	precision	–	–	0.9900±1.1e-2	0.9967±1.5e-3
	recall	–	–	0.9921±2.8e-3	0.9883±2.4e-2
	f1-score	–	–	0.9910±4.3e-3	0.9924±1.2e-2
uk	precision	–	0.9193±1.9e-3	0.9995±3.8e-4	0.9996±4.2e-4
	recall	–	0.9978±6.5e-4	0.9968±4.5e-3	0.9986±8.0e-4
	f1-score	–	0.9569±9.7e-4	0.9981±2.1e-3	0.9991±4.8e-4
uz	precision	–	–	0.9978±2.5e-3	0.9931±3.5e-3
	recall	–	–	0.9945±9.8e-4	0.9938±7.6e-3
	f1-score	–	–	0.9962±1.5e-3	0.9934±3.7e-3

Таблица 1: Сравнение показателей ldig и lang-detect

	be	**ru**	**uk**
az	0	5	0
be	74533	302	20
hy	81	194	13
ka	9	83	5
kk	54771	13507	6454
ky	12962	60143	4
ro	0	1	0
ru	7	74701	11
tg	36	13536	71
tk	0	61	0
uk	10	105	74828
uz	0	90	0

Таблица 2: Матрица ошибок для lang-detect с использованием стандартных профилей

язык	**lang-detect (станд. проф.)**	**lang-detect (натрен. проф.)**
be	74560	74648
ro	74986	74975
ru	74812	74770
uk	74817	74889

Таблица 3: Сравнение количества верно определенных представителей для lang-detect с использованием различных профилей

5. Эксперименты на пользовательском контенте

5.1. Обработка исходных данных

Ранее было рассмотрено поведение моделей на подготовленном корпусе. Корпус был собран на основе Википедии и содержит грамматически корректные сообщения. Однако часто можно столкнуться с иной ситуацией, когда созданный пользователями контент имеет особую авторскую орфографию. Подобная ситуация характерна для социальных сетей.

Работа полученных моделей была протестирована на Odnoklassniki Likes Dataset [4]. Данный датасет содержит сообщения, опубликованные в социальной сети «Одноклассники». Контент представлен в виде файлов **train_content.csv test_content.csv**, имеющих следующую структуру:

- **group_id** – анонимизированный идентификатор группы, содержащей сообщение
- **post_id** – анонимизированный идентификатор сообщения
- **timestamp** – время публикации сообщения
- **content** – содержание сообщения

В данной работе нас интересует текстовый контент, поэтому были рассмотрены только непосредственно сами сообщения, содержащиеся в файле **train_content.csv**.

Сообщения опубликованы пользователями и содержат множество ошибок, опечаток, гиперссылок и зашумленностей другого вида. Кроме того в содержание сообщений входят системные слова для обозначения опросов и изображений. Поэтому необходима предварительная обработка исходных данных. На данном этапе из текста исключаются системные слова, числа, гиперссылки и различные специальные символы.

Важным моментом является размер сообщений. Длина сообщений в датасете варьируется, однако нас преимущественно интересуют сообщения короткой длины, ограниченные в 140 символов. Поэтому для тестирования работы моделей из сообщений, превышающих указанную длину, извлекались подстроки от начала сообщения до конца вхождения последнего слова, не превышая необходимой длины.

5.2. Результаты на пользовательском контенте

Оценить результат работы на всем датасете весьма затруднительно ввиду его размера. Основная сложность в том, что сообщения не

содержат меток о языке. Известно, что большая часть контента написана на русском, однако также присутствуют украинский, грузинский, армянский, узбекский и другие языки. Для того, чтобы приблизительно оценить распределение языков, было вручную рассмотрено двести случайных сообщений. Из них на русском языке оказались 182, на узбекском — 7, на азербайджанском — 3, на грузинском — 2, на армянском — 2, на украинском — 2, на киргизском — 1, на таджикском — 1. Следует заметить, что здесь и далее двуязычные сообщения, состоящие из русского и какого-либо другого языка, были отнесены к классу этого другого языка.

Далее была протестирована работа ldig и lang-detect. Работа lang-detect была рассмотрена сначала с использованием натренированных профилей, а затем с использованием стандартных при добавлении недостающих натренированных. Как и ранее, ввиду размеров датасета, было рассмотрено по 200 сообщений, отнесенных в класс соответствующего языка, и вручную оценена правильность принадлежности к классу. В таблице 4 отражено количество верно проставленных меток для языков, имеющих не менее двух представителей.

язык	**ldig**	**lang-detect**	**lang-detect (станд. проф.)**
az	145	166	–
hy	92	199	–
ka	93	197	–
ru	198	199	197
uk	58	77	78
uz	173	125	–

Таблица 4: Количество верно определенных сообщений для Odnoklassniki Likes Dataset

Из таблицы 4 можно сделать несколько выводов о работе моделей на данном датасете с использованием построенных профилей:

- для грузинского, армянского и украинского языков видно превосходство lang-detect

- для русского языка работа моделей оказалась примерно одинаковой
- для узбекского языка более качественной оказалось работа ldig
- для русского и украинского языков lang-detect показал схожие результаты как с использованием натренированных профилей, так и с использованием стандартных

Заключение

В данной работе были рассмотрены основные методы определения языка в коротких текстах. Были получены следующие результаты:

- подготовлен корпус языков ближнего зарубежья для построения языковых профилей
- построены языковые профили на основе N-грамм и ∞-грамм
- проведен сравнительный анализ подходов на подготовленном корпусе и реальном пользовательском контенте Odnoklassniki Likes Dataset [4]

Сравнительный анализ показал преимущество подхода с использованием N-грамм над подходом, основывающемся на использовании ∞-грамм, а также выявил необходимость добавления дополнительных языковых профилей для повышения качества результатов.

Полученные в ходе работы корпусы и языковые профили выложены в открытый доступ. [1]

[1] `https://github.com/denniean/language_profiles`

Список литературы

[1] Cavnar William B, Trenkle John M et al. N-gram-based text categorization // Ann Arbor MI. — 1994. — Vol. 48113, no. 2. — P. 161–175.

[2] Grefenstette Gregory. Comparing two language identification schemes // 3rd International conference on Statistical Analysis of Textual Data. — 1995.

[3] Natural language identification using corpus-based models / Clive Souter, Gavin Churcher, Judith Hayes et al. // Hermes journal of linguistics. — 1994. — Vol. 13, no. S 183. — P. 203.

[4] Odnoklassniki Likes Dataset. — URL: `http://likesdataset.sh2014.org/`.

[5] Okanohara Daisuke, Tsujii Jun'ichi. Text Categorization with All Substring Features. // SDM. — 2009. — P. 838–846.

[6] Prager John M. Linguini: Language identification for multilingual documents // Journal of Management Information Systems. — 1999. — Vol. 16, no. 3. — P. 71–101.

[7] Shuyo Nakatani. Language Detection Library for Java. — 2010. — URL: `https://github.com/shuyo/language-detection`.

[8] Shuyo Nakatani. Short text language detection with infinity-gram. — 2012. — URL: `https://github.com/shuyo/ldig`.